Fredholm and Volterra Integral Equations of the Second Kind

William H. Press, and Saul A. Teukolsky

Citation: Computers in Physics **4**, 554 (1990); doi: 10.1063/1.4822946
View online: https://doi.org/10.1063/1.4822946
View Table of Contents: http://aip.scitation.org/toc/cip/4/5
Published by the American Institute of Physics

NUMERICAL RECIPES

Fredholm And Volterra Integral Equations Of The Second Kind

William H. Press and Saul A. Teukolsky

Integral equations are often the best way to formulate physics problems. However, the typical physics student gets almost no training in integral equations, in contrast to differential equations, for example. Many physicists thus believe that numerical solution of integral equations must be an extremely arcane topic, since it is almost never dealt with in numerical analysis textbooks!

Actually there is a voluminous and ever-growing literature on the numerical solution of integral equations, and recently several monographs have appeared.[1–3] One reason for all this activity is that there are many different kinds of equations, each with many different possible pitfalls, and often many different algorithms have been proposed to deal with just a single case. However, for a class of commonly occurring problems—Fredholm and Volterra equations of the second kind—it turns out that simple strategies are competitive with the more complicated algorithms that have recently been proposed.

Recall how integral equations are classified. Fredholm equations involve definite integrals:

$$g(t) = \int_a^b K(t,s)f(s)ds \tag{1}$$

is a Fredholm equation of the first kind, while

$$f(t) = \lambda \int_a^b K(t,s)f(s)ds + g(t) \tag{2}$$

is an equation of the second kind. Here, $f(t)$ is the unknown function, while $g(t)$ and $K(t,s)$ are given. $K(t,s)$ is called the *kernel*. Equation (2) is *inhomogeneous*; if $g(t) = 0$ then it is *homogeneous*, sometimes called an equation of the third kind. The parameter λ enters into theorems on the solvability of equation (2): For a bounded kernel, the homogeneous equation has solutions for at most a denumerably infinite set $\lambda = \lambda_n$, $n = 1,2,...$, the *eigenvalues* The corresponding solutions $f_n(t)$ are the *eigenfunctions*. The eigenvalues are real if the kernel is symmetric. The inhomogeneous equation has a solution except when λ is an eigenvalue (the *Fredholm alternative*).

In Volterra equations the upper limit of integration is the independent variable t. Equations of the first and second kinds are defined in the analogous way to Fredholm equations:

$$g(t) = \int_a^t K(t,s)f(s)ds \tag{3}$$

is a Volterra equation of the first kind, while

$$f(t) = \int_a^t K(t,s)f(s)ds + g(t) \tag{4}$$

is an equation of the second kind. Note that we omit the parameter λ in equation (4) since in the homogeneous case $g(t) = 0$, Volterra equations of the second kind with bounded kernels have no eigenvalues with square-integrable eigenfunctions.

We have specialized our definitions (1)–(4) to the case of linear integral equations. The integrand in a nonlinear version of equation (1) or (3) would be $K(t,s,f(s))$ instead of $K(t,s)f(s)$; a nonlinear version of equation (2) or (4) would have an integrand $K(t,s,f(t),f(s))$. Nonlinear Fredholm equations are considerably more complicated than their linear counterparts. Fortunately, they do not occur as frequently in practice and we shall ignore them in this column. By contrast, solving nonlinear Volterra equations usually involves only a slight modification of the algorithm for linear equations, as we shall see.

Fredholm equations of the first kind are usually extremely ill-conditioned. Applying the kernel to a function is generally a smoothing operation, so the solution, which requires inverting the operator, will be extremely sensitive to small changes or errors in the input. Specialized methods have been developed for such equations; we will return to them in a later column. Voltera equations of the first kind tend not to be as sensitive; the upper limit to the integral introduces a sharp step that nicely spoils the operator's smoothing. As a result, one can often get away with a simple algorithm, like the one we will describe below, for equations of the second kind.

Let us start by considering the numerical solution of equation (2). The basic method we shall describe is called the *Nystrom method*. It depends on the choice of some approximate *quadrature rule*:

$$\int_a^b y(s)ds = \sum_{j=1}^{N} w_j y(s_j). \tag{5}$$

Here, the set $\{w_j\}$ are the weights of the quadrature rule, while the N points $\{s_j\}$ are the abscissas. If we apply the quadrature rule (5) to equation (2), we get

$$f(t) = \lambda \sum_{j=1}^{N} w_j K(t,s_j)f(s_j) + g(t). \tag{6}$$

William H. Press is a professor of astronomy and physics at Harvard University. Saul A. Teukolsky is a professor of physics and astronomy at Cornell University.

Evaluate equation (6) at the quadrature points:

$$f(t_i) = \lambda \sum_{j=1}^{N} w_j K(t_i,s_j) f(s_j) + g(t_i). \qquad (7)$$

Let f_i be the vector $f(t_i)$, g_i the vector $g(t_i)$, K_{ij} the matrix $K(t_i,s_j)$, and $\tilde{K}_{ij} = K_{ij} w_j$. Then in matrix notation equation (7) becomes

$$(\mathbf{1} - \lambda \tilde{\mathbf{K}})\mathbf{f} = \mathbf{g}. \qquad (8)$$

This is a set of N linear algebraic equations in N unknowns which can be solved by standard Gaussian elimination techniques.[4–6] The solution is usually well-conditioned, unless λ is very close to an eigenvalue.

What quadrature rule should you use? It is certainly possible to solve integral equations with low-order quadrature rules like the repeated trapezoidal or Simpson's rules. However, the solution of the algebraic equations involves $O(N^3)$ operations, and so the most efficient methods tend to use high-order quadrature rules to keep N as small as possible. For smooth, nonsingular problems, nothing beats Gaussian quadrature[4–6] (Gauss–Legendre quadrature).

Having obtained the solution at the quadrature points $\{t_{ji}\}$, how do you get the solution at some other point t? You do *not* simply use polynomial interpolation. This destroys all the accuracy you have worked so hard to achieve. Nystrom's key observation was that you should use equation (6) as an interpolatory formula, maintaining the accuracy of the solution.

In Box 1 we give two subroutines for use with linear Fredholm equations of the second kind. fred2 sets up equation (8) and then solves it by LU decomposition with calls to the routines ludcmp and lubksb.[4–6] Of course you could substitute any other linear equation solvers for these calls. The Gauss–Legendre quadrature is implemented by first getting the weights and abscissas with a call to gauleg.[4–6] fred2 requires that you provide an external function that returns $g(t)$ and another that returns λK_{ij}. It then returns the solution f at the quadrature points. It also returns the quadrature points and weights. These are used by the second routine fredin to carry out the Nystrom interpolation of equation (6) and return the value of f at any point in the interval $[a,b]$.

One disadvantage of a method based on Gaussian quadrature is that there is no simple way to obtain an estimate of the error in the result. The best practical method is to increase N by 50%, say, and treat the difference between the two estimates as a conservative estimate of the error in the result obtained with the larger value of N.

Delves and Mohamed[1] have tested several different kinds of routines for straightforward Fredholm equations of the second kind. They concluded "...the clear winner of this contest has been the Nystrom routine...with the N-point Gauss–Legendre rule. This routine is extremely simple...Such results are enough to make a numerical analyst weep."

Many integral equations have singularities in either the kernel or the solution or both. A simple quadrature method will show poor convergence with N if such singularities are ignored. There is sometimes art in how singularities are best handled. Here, we offer a few suggestions:

(i) Integrable singularities can often be removed by a change of variable. For example, the singular behavior $K(t,s) \sim s^{1/2}$ or $s^{-1/2}$ near $s = 0$ can be removed by the transformation $z = s^{1/2}$. [We are assuming that the singular behavior is confined to K, whereas the quadrature actually involves the product $K(t,s)f(s)$ and it is this product that must be "fixed." Hopefully, you can deduce the singular nature of the product before you try a numerical solution, and take the appropriate action.]

(ii) If $K(t,s)$ can be factored as $w(s)\tilde{K}(t,c)$ where $w(s)$ is singular and $\tilde{K}(t,s)$ is smooth, then a Gaussian quadrature based on $w(s)$ as a weight function will work well. Even if the factorization is only approximate, the convergence is often improved dramatically. All you have

Box 1.

```
      SUBROUTINE fred2(n,a,b,t,f,w,g,ak)
      INTEGER n,NMAX
      REAL a,b,f(n),t(n),w(n),ak,g
      PARAMETER(NMAX=200)
C     USES ak,g,gauleg,lubksb,ludcmp
        Solves a linear Fredholm equation of the second kind. On input, a and b are the limits of
        integration, and n is the number of points to use in the Gaussian quadrature. g and ak
        are user-supplied external functions that respectively return g(t) and λK(t,s). The routine
        returns arrays t(1:n) and f(1:n) containing the abscissas t_i of the Gaussian quadrature and
        the solution f at these abscissas. Also returned is the array w(1:n) of Gaussian weights for
        use with the Nystrom interpolation routine fredin.
      INTEGER i,j,indx(NMAX)
      REAL d,omk(NMAX,NMAX)
      if(n.gt.NMAX)pause 'increase NMAX'
      call gauleg(a,b,t,w,n)                    Replace gauleg with another routine if not using
                                                  Gauss-Legendre quadrature.
      do 12 i=1,n
        do 11 j=1,n                             Form 1 - λK̃.
          if(i.eq.j)then
            omk(i,j)=1.
          else
            omk(i,j)=0.
          endif
          omk(i,j)=omk(i,j)-ak(t(i),t(j))*w(j)
        enddo 11
        f(i)=g(t(i))
      enddo 12
      call ludcmp(omk,n,NMAX,indx,d)            Solve linear equations.
      call lubksb(omk,n,NMAX,indx,f)
      return
      END
```

Box 2.

```
      FUNCTION fredin(x,n,a,b,t,f,w,g,ak)
      INTEGER n
      REAL fredin,a,b,x,f(n),t(n),w(n),ak,g
C     USES ak,g
        Given arrays t(1:n) and w(1:n) containing the abscissas and weights of the Gaussian quadra-
        ture, and given the solution array f(1:n) from fred2, this function returns the value of f
        at x using the Nystrom interpolation formula. On input, a and b are the limits of integration,
        and n is the number of points used in the Gaussian quadrature. g and ak are user-supplied
        external functions that respectively return g(t) and λK(t,s).
      INTEGER i
      REAL sum
      sum=0.
      do 11 i=1,n
        sum=sum+ak(x,t(i))*w(i)*f(i)
      enddo 11
      fredin=g(x)+sum
      return
      END
```

to do is replace gauleg in the routine fred2 by another quadrature routine. Our last column[7] explained how to construct such quadratures; or you can find tabulated abscissas and weights in the standard reference.[8,9] You must also supply $\tilde{K}$ instead of K. This method is a special case of the *product Nystrom method*,[1,3] where one factors out a singular term $p(t,s)$ depending on both t and s from K and constructs suitable weights for a Gaussian quadrature. The calculations in the general case are quite cumbersome, because the weights depend on the chosen $\{t_i\}$ as well as the form of $p(t,s)$. In practice, it has been used only with the repeated trapezoidal or Simpson's rules, since then the quadratures involve only 2 or 3 points.

(iii) An infinite range of integration is also a form of singularity. Truncating the range at a large finite value should only be used as a last resort. If the kernel goes rapidly to zero, then a Gauss–Laguerre $[w\sim\exp(-\alpha s)]$or Gauss–Hermite $[w\sim\exp(-s^2)]$ quadrature should work well. Long-tailed functions often succumb to the transformation

$$s=2\alpha/(z+1)-\alpha,$$

which maps $[0,\infty]$ to $[-1,1]$ so that Gauss–Legendre integration can be used. Here, $\alpha>0$ is a constant you can adjust to improve the convergence.

(iv) A common situation in practice is that $K(t,s)$ is singular along the line $t=s$. Here, the Nystrom method fails completely because the kernel gets evaluated at (t_i,s_i).

Subtraction of the singularity is the cure:

$$\begin{aligned}\int_a^b K(t,s)f(s)ds &= \int_a^b K(t,s)[f(s)-f(t)]ds+\int_a^b K(t,s)f(t)ds\\ &= \int_a^b K(t,s)[f(s)-f(t)]ds+r(t)f(t),\end{aligned}$$

where $r(t)=\int_a^b K(t,s)ds$ is computed analytically or numerically. If the first term on the right-hand side is now regular, we can use the Nystrom method. Instead of equation (7), we get

$$f_i=\lambda\sum_{\substack{j=1\\ j\neq i}}^{N} w_jK_{ij}[f_j-f_i]+\lambda r_i f_i+g_i. \tag{9}$$

Sometimes the subtraction process must be repeated before the kernel is completely regularized. See Ref. 1 for details.

Turn now to solutions of the homogeneous equation. If we set $\lambda=1/\mu$ and $\mathbf{g}=0$, then equation (8) becomes a standard eigenvalue equation

$$\tilde{\mathbf{K}}\mathbf{f}=\mu\mathbf{f},$$

which we can solve with any convenient matrix eigenvalue routine.[4–6] Note that if our original problem had a symmetric kernel, then the matrix $\mathbf{K}$ is symmetric. However, since the weights w_j are not equal for most quadrature rules, the matrix $\tilde{\mathbf{K}}$ is not symmetric. The matrix eigenvalue problem is much easier for symmetric matrices, and so we should restore the symmetry if possible. Provided the weights are positive (which they are for Gaussian quadrature), we can define the diagonal matrix $\mathbf{D}=\mathrm{diag}(w_j)$ and its square root, $\mathbf{D}^{1/2}=\mathrm{diag}(\sqrt{w_j})$. Then equation (10) becomes

$$\mathbf{KDf}=\mu\mathbf{f}.$$

Multiplying by $\mathbf{D}^{1/2}$, we get

$$(\mathbf{D}^{1/2}\mathbf{K}\mathbf{D}^{1/2})\mathbf{h}=\mu\mathbf{h}, \tag{11}$$

where $\mathbf{h}=\mathbf{D}^{1/2}\mathbf{f}$. Equation (11) is now in the form of a symmetric eigenvalue problem.

Solution of equation (10) or (11) will in general give N eigenvalues, where N is the number of quadrature points used. For square-integrable kernels, these will provide good approximations to the lowest N eigenvalues of the integral equation. Kernels of *finite rank* (also called *degenerate* or *separable* kernels) have only a finite number of eigenvalues (possibly none). You can diagnose this situation by a cluster of eigenvalues μ that are zero to machine precision. The number of nonzero eigenvalues will stay constant as you increase N to improve their accuracy. Some care is required here: A nondegenerate kernel has an infinite number of eigenvalues that have an accumulation point at $\mu=0$. You distinguish the two cases by the behavior of the solution as you increase N. If you suspect a degenerate kernel, you will usually be able to solve the problem by analytic techniques described in all the textbooks.

Let's now look at Volterra equations, of which our prototype is equation (4). Most algorithms for Volterra equations march out from $t=a$, building up the solution as they go along. In this sense they resemble initial value problems for ordinary differential equations (ODE), and many of the algorithms for ODE's have their counterparts for Volterra equations. The simplest way to proceed is to solve the equation on a mesh with uniform spacing:

$$t_i=a+ih,\quad i=0,1,\ldots,N,\quad h\equiv(b-a)/N. \tag{12}$$

Again, we choose a quadrature rule. For a uniform mesh, the simplest scheme is the trapezoidal rule:

$$\int_a^{t_i} K(t_i,s)f(s)ds = h\left(\frac{1}{2}K_{i0}f_0+\sum_{j=1}^{i-1}K_{ij}f_j+\frac{1}{2}K_{ii}f_i\right). \tag{13}$$

Thus the trapezoidal method for equation (4) is

$$\left(1-\frac{1}{2}hK_{ii}\right)f_i=h\left(\frac{1}{2}K_{i0}f_0+\sum_{j=1}^{i-1}K_{ij}f_j\right)+g_i,\quad i=1,\ldots,N. \tag{14}$$

Equation (14) is an explicit prescription that gives the solution in $O(N^2)$ operations—Volterra equations usually

Box 3.

```
      SUBROUTINE vol2ln(n,m,t0,h,t,f,g,ak)
      INTEGER m,n,MMAX
      REAL h,t0,f(m,n),t(n),ak,g
      PARAMETER(MMAX=5)
C     USES ak,g,lubksb,ludcmp
        Solves a set of m linear Volterra equations of the second kind using the extended trapezoidal
      rule. On input, t0 is the starting point of the integration and n-1 is the number of steps
      of size h to be taken. g(k,t) is a user-supplied external function that returns g_k(t), while
      ak(k,l,t,s) is another user-supplied external function that returns the (k,l) element of the
      matrix K(t,s). The solution is returned in f(1:m,1:n), with the corresponding abscissas in
      t(1:n).
      INTEGER i,j,k,l,indx(MMAX)
      REAL d,sum,a(MMAX,MMAX),b(MMAX)
      t(1)=t0
      do11 k=1,m                                  Initialize.
        f(k,1)=g(k,t(1))
      enddo11
      do16 i=2,n                                  Take a step h.
        t(i)=t(i-1)+h
        do14 k=1,m
          sum=g(k,t(i))                           Accumulate right-hand side of linear equations
          do13 l=1,m                                in sum.
            sum=sum+0.5*h*ak(k,l,t(i),t(1))*f(l,1)
            do12 j=2,i-1
              sum=sum+h*ak(k,l,t(i),t(j))*f(l,j)
            enddo12
            if(k.eq.l)then                        Left-hand side goes in matrix a.
              a(k,l)=1.
            else
              a(k,l)=0.
            endif
            a(k,l)=a(k,l)-0.5*h*ak(k,l,t(i),t(i))
          enddo13
          b(k)=sum
        enddo14
        call ludcmp(a,m,MMAX,indx,d)              Solve linear equations.
        call lubksb(a,m,MMAX,indx,b)
        do15 k=1,m
          f(k,i)=b(k)
        enddo15
      enddo16
      return
      END
```

involve less work than the corresponding Fredholm equations. This is somewhat counterbalanced by the fact that *systems* of Volterra equations occur more frequently in practice. If we interpret equation (4) as a *vector* equation for the vector of m functions $f(t)$, then the kernel $K(t,s)$ is an $m \times m$ matrix. Equation (14) must now be understood also as a vector equation. For each i, we have to solve the $m \times m$ set of linear algebraic equations by Gaussian elimination. The routine vol2ln in Box 3 implements this algorithm. You must supply an external function that returns the k th function of the vector $g(t)$ at the point t, and another that returns the (k,l) element of the matrix $K(t,s)$ at (t,s). vol2ln then returns the vector $f(t)$ at the regularly spaced points t_i.

For nonlinear Volterra equations, equation (14) holds with the product $K_{ii} f_i$ replaced by $K_{ii}(f_i)$, and similarly for the other two products of K's and f's. Thus for each i we solve a nonlinear equation for f_i with a known right-hand side. Newton's method with an initial guess of f_{i-1} usually works very well provided the step size is not too big.

Higher-order methods for solving Volterra equations are, in our opinion, not as important as for Fredholm equations since Volterra equations are relatively easy to solve. However, there is an extensive literature on the subject. Several difficulties arise. First, any method that achieves higher order by operating on several quadrature points simultaneously will need a special method to get started, when values at the first few points are not yet known. Second, stable quadrature rules can give rise to unexpected instabilities in integral equations. For example, suppose we try to replace the trapezoidal rule in the algorithm above with Simpson's rule. Simpon's rule naturally integrates over an interval $2h$, so we easily get the function values at the even mesh points. For the odd mesh points, we could try appending one panel of trapezoidal rule. But to which end of the integration should we append it? We could do one step of trapezoidal rule followed by all Simpson's rule, or Simpson's rule with one step of trapezoidal rule at the end. Surprisingly, the former scheme is unstable, while the latter is fine!

A simple approach that can be used with the trapezoidal method given above is Richardson extrapolation: Compute the solution with step size h and $h/2$. Then, assuming the error scales with h^2, compute

$$f_E = [4f(h/2) - f(h)]/3.$$

This procedure can be repeated as with Romberg integration.

The general consensus[1,2] is that the best of the higher-order methods is the "block-by-block" method. Another important topic is the use of variable step size methods, which are much more efficient if there are sharp features in K or f. Variable step size methods are quite a bit more complicated than their counterparts for differential equations; we refer you to the literature[1,2] for a discussion.

Finally, we need to remind you to be on the lookout for singularities in the integrand. Product integration works well with the trapezoidal rule provided you can factor out an analytically integrable singular piece from the integrand. ■

In our next column: Savitzky–Golay filters.

References

1. L. M. Delves and J. L. Mohamed, *Computational Methods for Integral Equations* (Cambridge U. P., New York, 1985).
2. P. Linz, *Analytical and Numerical Methods for Volterra Equations* (SIAM, Philadelphia, 1985).
3. K. E. Atkinson, *A Survey of Numerical Methods for the Solution of Fredholm Integral Equations of the Second Kind* (SIAM, Philadelphia, 1976).
4. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes: The Art of Scientific Computing* (Cambridge U. P., New York, 1986).
5. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing* (Cambridge U. P., New York, 1988).
6. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in Pascal: The Art of Scientific Computing* (Cambridge U. P., New York, 1988).
7. W. H. Press and S. A. Teukolsky, Comput. Phys. **4**(4), 423 (1990).
8. M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions* (National Bureau of Standards, Washington, 1964; reprinted by Dover Publications, New York, 1968).
9. A. H. Stroud and D. Secrest, *Gaussian Quadrature Formulas* (Prentice-Hall, Englewood Cliffs, NJ, 1966).